Glimpse: A Programmable Early-Discard Camera Architecture for Continuous Mobile Vision

Saman Naderiparizi
Microsoft
U. of Washington

Pengyu Zhang
Microsoft
U. of Massachusetts Amherst

Matthai Philipose
Microsoft

Bodhi Priyantha
Microsoft

Jie Liu
Microsoft

Deepak Ganesan
U. of Massachusetts Amherst

ABSTRACT

We consider the problem of continuous computer-vision based analysis of video streams from mobile cameras over extended periods. Given high computational demands, general visual processing must currently be offloaded to the cloud. To reduce mobile battery and bandwidth consumption, recent proposals offload only "interesting" video frames, discarding the rest. However, determining what to discard is itself typically a power-hungry computer vision calculation, very often well beyond what most mobile devices can afford on a continuous basis. We present the Glimpse system, a redesign of the conventional mobile video processing pipeline to support such "early discard" flexibly, efficiently and accurately. Glimpse is a novel architecture that gates wearable vision using low-power vision modalities. Our proposed architecture adds novel sensing, processing, algorithmic and programming-system components to the camera pipeline to this end. We present a complete implementation and evaluation of our design. In common settings, Glimpse reduces mobile power and data usage by more than one order of magnitude relative to earlier designs, and moves continuous vision on lightweight wearables to the realm of the practical.

Keywords

Continuous mobile vision; low-power early discard; low-power vision modalities; energy efficient wearable vision system

1. INTRODUCTION

Continuous visual analysis of video from wearable devices has the potential to enable many applications beyond those based on conventional wearable sensors such as accelerometers and GPS [4, 14, 24, 30, 43]. However, modern computer vision techniques, such as those based on Deep Neural Networks, have resource demands that require cloud-based servers rather than mobile processors [16]. The obvious solution, off-loading all computer vision to the cloud, drains the typical mobile battery within two hours, motivating a solution that can inexpensively select a few frames for cloud processing. To our knowledge, no mobile-battery-based solution to date is able to run continuously for a 10-hour day while allowing flexible selection of informative frames. In this paper, we present Glimpse, a hardware-software add-on to the traditional mobile imaging pipeline that allows frames to be selected for processing *efficiently*, averaging under 100mW over a 10-hour day, and *flexibly*, allowing safe application-level extension.

For continuous functioning, a frame-selection subsystem must address the *resource-accuracy tradeoff* between resource usage, i.e., restricting the execution cost of rejection so it can run for extended periods, and accuracy, i.e., rejecting as many uninteresting frames as possible while preserving the interesting ones. A traditional solution to this problem is to use low-power sensors such as inertial, light [21], time [22] and even skin conductance [19] to trigger frame selection. These techniques score high on the resource-usage axis, since these sensors usually consume orders of magnitude less power than the image processing pipeline. However, in terms

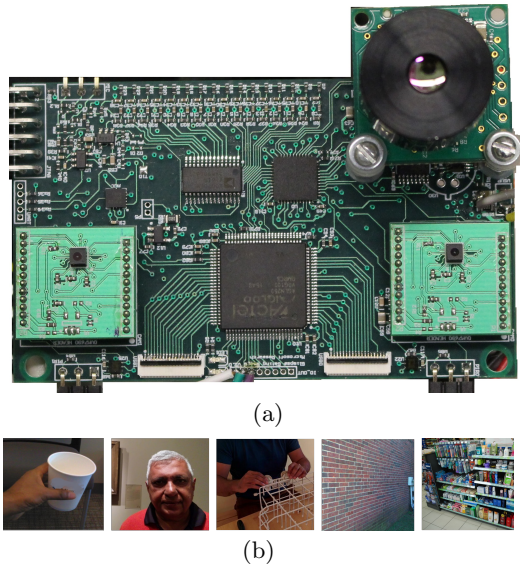


Figure 1: **Glimpse prototype and its applicability.**(a) Glimpse prototype board (size 5 × 8cm) (b) Entities of interest to applications: the wearer's hands [41], their conversational partners' faces [25], their partners' limbs [7], nearby walls [8, 47] and indoor places [44].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MobiSys'17, June 19-23, 2017, Niagara Falls, NY, USA

© 2017 ACM. ISBN 978-1-4503-4928-4/17/06…$15.00

DOI: http://dx.doi.org/10.1145/3081333.3081347

of accuracy, i.e. detecting all events of interest, they are limited: many situations of interest are simply not detectable using low-datarate sensors.

At the other end of the spectrum are approaches that use (less expensive) visual processing to trigger more expensive visual processing. Such vision-based triggers can be quite rich relative to sensors, and include, e.g., those based on pixel values (e.g., faces [25]), depth values (e.g., foreground [8]) or temporal variation (e.g., "change" detection [9]). However, because they re-use existing machinery (imagers, algorithms and processors) for visual processing, these approaches are limited by the design assumptions baked into these components:

- **Imagers** The quality of imaging necessary for selection is often much less then that for recognition. E.g., face detection works well at 25×25 grayscale [49] pixel resolution; recognition requires 200×200 color [40].
- **Algorithms** Particular selection tasks often require much less precision than general algorithms provide. E.g., detecting a camera-wearer's hand may only require checking if a pixel is within 1m, whereas a stereo depth system may return cm-level depth for every pixel.
- **Processors** Standard mobile application processors may be energy inefficient because of the mismatch between highly parallelizable image computations and the inherently sequential nature of CPUs. For example, temporal frame differencing for change detection can easily keep a mobile applications processor busy [9], impractical for day-long processing.

To avoid the overhead of reusing the standard vision pipeline while preserving the benefits of vision-based frame selection, we advocate that **vision-based frame selection be treated as a first-class design goal, with a corresponding dedicated hardware/software subsystem in a mobile device**. To this end, we present Glimpse, a system dedicated to discarding uninteresting frames by itself performing coarse visual processing at very low power. Glimpse combines several novel components in a novel way: An array of *gating sensors and imagers* (including a thermal imager and a stereo pair) draws substantially lower power than the primary imager, and is adequate for many selection (but possibly, not deeper analysis) tasks. A suite of *coarse image processing* algorithms is optimized for common early rejection sub-tasks using the gating imagers. A dedicated efficient *gating computation fabric*, comprising of a low-power microcontroller and an FPGA, with efficient access to the gating imagers, is designed to execute (coarse) vision algorithms more efficiently than the standard CPU/DRAM model. A simple programming model, based on *rejection cascades*, allows applications to combine the output of gating sensors into efficient, customized frame-selection classifiers.

Glimpse is fully implemented as a modular circuit board (Figure 1a) and embedded software. We tested its frame selection abilities by wearing it for 15 hours over 3 days and programming it to select frames where a new person entered the personal space of the wearer (as a front end for social assistance [25, 33]), and those where the wearer's hand came in the field of view (as a front end for activity tracking applications [41, 50]). Glimpse is able to detect frames representing events of interest over 87% (100%) of the time for visual events longer than 1s (3s) while drawing roughly 41-54mW for frame selection and 43-153mW in total. These represent a 10-20× improvement over state-of-the-art mobile application processor based implementations. We believe our work is a significant step toward day-long, programmable, continuous vision on wearable-device energy budgets.

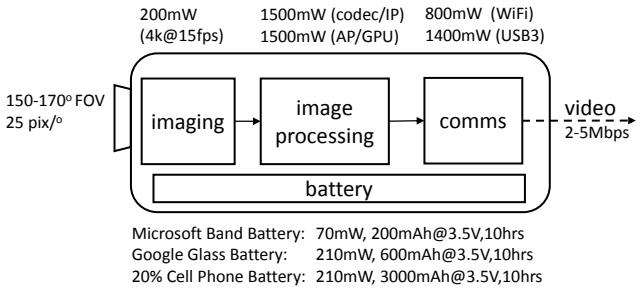


Figure 2: **Wearable camera pipeline.** Day-long continuous operation off a band/wearable-sized battery is infeasible with the conventional wearable camera systems pipeline.

2. BACKGROUND AND MOTIVATION

Figure 2 illustrates the structure and key performance characteristics of a wearable camera pipeline comprised of state-of-the art components. The basic pipeline consists of a high-resolution, large field of view color imager[1] feeding into a compression and/or image processing subsystem. The resulting video stream is either stored on board (not shown) or transmitted using either wireless or wired communications ("comms") module. In the figure, we detail the power *consumption* of each component above the camera, and also specify the average power *budget* of the entire camera, under varying battery assumptions, below the camera.

We now consider two configurations. In the "full offload" model, we could compress the resulting frames into a video stream and offload it to, e.g., the cloud. Compression, via a codec/image processor (IP) combination, would require at least another 1.5W on top of about 200mW high-resolution camera power consumption [1], and the resulting stream would require a steady 800mW or more to offload via WiFi (and considerably more via WWAN). In the "selective offload" model, we could use an application processor (AP) to detect and transmit frames of interest. Keeping up with high-resolution 15fps would easily require multiple cores, including GPU support [28], and consume 1.5W or more. In this latter model, we would hopefully need to transmit frames only a small fraction of the time (e.g., 10% of the time), so that transmission power in this case would be lower, say 10% of 800mW, say 80mW. In either model, average power draw from the camera would exceed 2W. These numbers are in line with recent reported mobile vision systems [9].

This level of power consumption is problematic for two reasons. First, as the lower part of Figure 2 shows, most reasonable battery configurations will only yield at most roughly 200mW. Thus, if the wearable device is to be powered by either a compact battery (a la Google Glass) or by a fraction of a cell-phone battery (a generous 20% of the battery), a constant power draw of 2W is roughly 10× more than is sustainable. Second, for wearable devices of the Glass

[1] A large field of view is critical to wearable devices because interesting events occur in a wide area in front of the user.

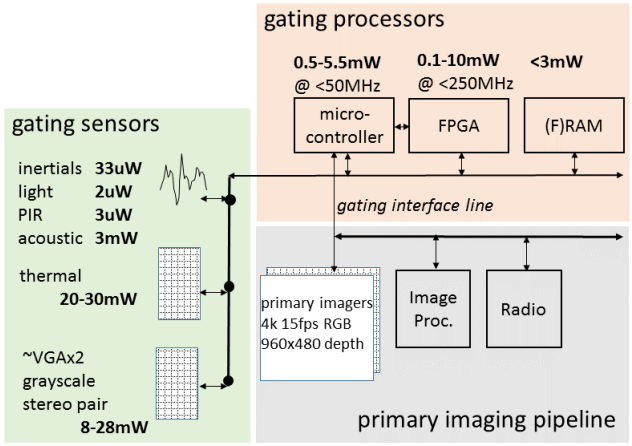


Figure 3: **The Glimpse hardware architecture.**

form factor, a reasonable rule of thumb is that each Watt of power dissipated raises the temperature of the device by 10°C [28] above its surroundings. The resulting 20°C (36°F) temperature rise is quite noticeable and potentially unpleasant for wearable devices. Thus even if it is feasible to power the entire camera via a wire connected to a large battery pack, e.g., police cameras (this is the "USB" transmit option in the figure), significantly reducing device power draw may be important.

3. SYSTEM ARCHITECTURE

We now present the hardware and software architecture of Glimpse.

3.1 Hardware Architecture

The primary imaging pipeline, as outlined in Section Section 2 and colored gray in Figure 3, will typically contain high-power imaging, computation and communications components that consume an order of magnitude more power than budgeted. Glimpse duty-cycles this pipeline so that aggregate power consumption is below budget, implying a duty cycle of 10% or less.

The design philosophy of Glimpse is to provide a variety of low-power sensing (crucially, including a variety of imaging options) and processing capabilities that can be combined in a flexible manner to select frames inexpensively for a given application. Glimpse therefore has two primary subsystems (green and orange in Figure 3). *Gating sensors* run at relatively high duty cycle compared to the primary imagers and must consume relatively little power. *Gating processors* initialize sensors, control communication with sensors, run application extensions on sensed data, and coordinate task execution with other processors.

Sensors typically run in the 1uW to tens of mW range. These sensors are selected such that, for many applications, they can (individually or collectively) predict whether the primary imagers are likely to detect events of interest to the application at any given time. Note that, crucially, gating sensors are not required to detect the interesting events themselves. Thus, for example:

The **inertial sensor** (accelerometer and gyroscope) may indicate that the velocity of the camera is high enough that motion blur will make it unlikely that faces, or other specific objects, can be detected or recognized.

The **light sensor** may indicate that light levels are not adequate for detection or recognition.

The **passive-infrared** sensor, which is triggered when the thermal signature of the field of view changes, may indicate that it is unlikely that people or their parts are close to the camera in the field of view.

The **microphone**, with accompanying custom analog spectral decomposer for sub-mW acoustic feature extraction, may indicate, based on acoustic activity, that the wearer is in an uninteresting activity state (e.g., driving), or social context (e.g., not in conversation).

The **thermal (or far-infrared/FIR) imager**, which reports the temperature of every pixel in the field of view at 9 fps, may identify the *parts* of the field of view that are unlikely to contain people, vehicles, lights, monitors, etc as per their temperature.

The **VGA grayscale imager pair**, a pair of synchronized low-resolution grayscale imagers, from which a depth map of the field of view may be derived, may indicate that a part or whole of the field of view is too far/close to be of interest.

In every case, these sensors act as gatekeepers to the primary imagers, hence the term "gating" sensors. Good gating sensors have a few key characteristics:

1. Low power draw relative to the primary imager.
2. High information content in terms of predicting, for a variety of applications, the required behavior of the primary imaging pipeline *with high recall*: the gating sensor must seldom falsely imply that the primary imager need not wake up.
3. Low latency in making the prediction, since the primary imagers then need to be turned on to analyze the detected image.

The gating calculations themselves, which determine if a certain application is likely to find the current primary-imager frame interesting, are performed on the *gating processors*. These calculations range from simple digital IO checks (e.g., on motion detection using passive IR sensors) to lightweight computer vision algorithms (e.g., depth from stereo on the grayscale pair and object tracking on a thermal image). Gating processors must run at substantially lower power than the primary imaging pipeline. Glimpse achieves this by providing a very low-power microcontroller for simple calculations and user-friendly interface and a low power FPGA to handle the vision algorithms, with a small amount (e.g., 1 MB) of static RAM for buffering. The FPGA is also responsible for reading data from gating imagers, which allows FPGA-based vision algorithms to process pixels in a "streaming" manner as they are read off the imager *with no intermediate memory accesses*.

In summary, two aspects of the Glimpse hardware architecture are especially worth noting. First, it advocates the use of multiple possibly unconventional imaging technologies (e.g., thermal imager, low-power grayscale imager and stereo pair) dedicated to frame selection. Second, it incorporates dedicated programmable hardware (i.e., FPGAs) tightly coupled with these imagers for computations related to selection.

3.2 Software Architecture

The Glimpse software architecture is designed to allow applications to specify custom frame selection criteria in a sim-

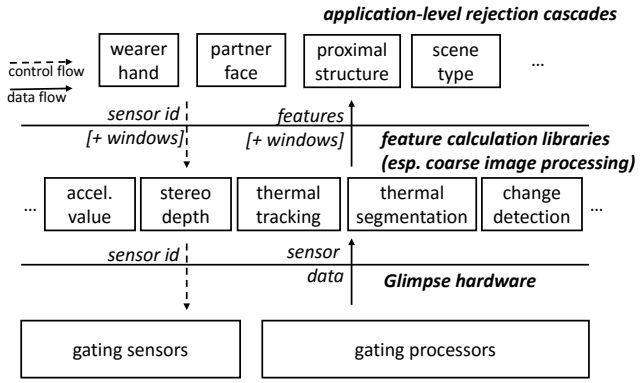


Figure 4: **The Glimpse software architecture.**

ple, expressive and safe manner. Several applications should be able to simultaneously register interest in frames, based on a rich range of conditions without interfering with each other, and without having to write Glimpse microcontroller or FPGA code.

Glimpse takes the perspective that frame selectors are essentially *cost-sensitive classification cascades* [49, 51]. As such, they can be divided into relatively heavyweight *feature calculation libraries* and lightweight *rejection cascades* (Figure 4), which applications can safely chain together to make increasingly expressive but efficient classifiers as described below.

3.2.1 Feature calculation libraries

Feature calculation is performed exclusively by optimized native libraries. Features range from simple wrappers around sensors like accelerometers to versions of visual primitives such as depth estimation, tracking, segmentation and change detection. Only feature calculators are allowed to interact directly with sensors. Feature calculation is optimized both via the design of new "coarse" versions of vision algorithms (discussed in Section 4), and also via careful implementation of these algorithms that, for instance, exploits the underlying hardware such as the FPGA.

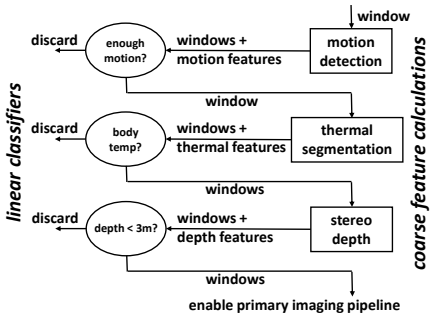
Each feature calculation function is required to take as input a set of windows partitioning the field of view of the primary imager and return a set of windows, each annotated with a vector of values. For instance the "accelerometer" value feature calculator may ignore the contents of its input window and simply annotate it with system acceleration. On the other hand, the thermal segmentation function, given a window, returns a set of sub-windows along with the temperature for each window. The depth estimator annotates each of its input windows with the (coarse) depth of that window.

3.2.2 Interpreting rejection cascades

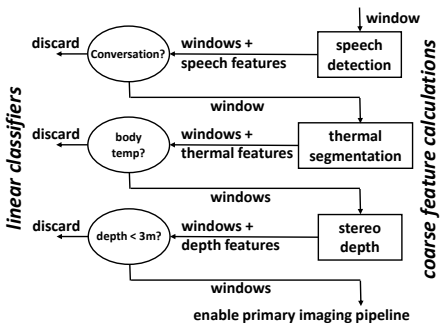
Applications are allowed to define *rejection cascades*, small programs that define classifiers. Glimpse provides a simple embedded *cascade interpreter* to execute the cascade and notify the primary imaging pipeline.

A cascade is a list of triples $(f_i, \hat{w}_i, t_i)$. Each triple consists of a reference f to a feature calculator, a vector $\hat{w}$ of linear-classification weights and a single scalar threshold value t. Given a set X_{i-1} of windows from the previous step of the cascade (the first step always starts with a single window that encompasses the entire field of view of the primary imager), the interpreter applies feature calculator f_i to produce a new set $X'_i = f(X_{i-1})$ of candidate windows and corresponding feature vectors V_i. For each candidate window $x'_{ij} \in X'_i$ and corresponding feature vector $\hat{v}_{ij} \in V_i$, the interpreter includes window x'_{ij} in set X_i for processing in the next cascade step if $t_i > \hat{v}_{ij}\hat{w}_i$, and rejects it otherwise. If, at any step, no windows remain to be forwarded, the interpreter stops. After the final stage of the cascade, if any (non-rejected) windows remain, the interpreter notifies the primary imaging pipeline to select a frame.

In Figure 4, solid arrows represent data flow, and dashed lines control flow, during interpreter execution.



(a) Human + hand detection



(b) Conversation partner detection

Figure 5: **Sample cascades.** 5a shows a cascade that detetcts if the wearer is either interacting with a human or with an object in his hand. 5b shows a cascade that triggers whenever the wearer is in conversation with another person.

3.2.3 Sample cascades

Figure 5 illustrates two rejection cascades. Figure 5a shows a cascade that triggers whenever either the wearer is interacting with a human or an object in his hand. This requires existence of human body parts closer than a certain distance (here set to 3m) from the wearer. It checks a motion-detector (typically passive infrared based) to check for adequate thermal activity in the field of view (either due to background or the wearer motion). If not, it discards the frame. If so, it uses thermal imager output to look for sub-windows in the frame that are at body temperature; if none are found, it rejects the frame. Finally, it checks the depth of the surviving sub-windows, retains any within 3 meters of distance and rejects the rest. If any windows survive, the primary imaging pipeline is enabled. Feature extractors that are inexpensive are expected earlier in the cascade, and more informative, and usually more expensive ones, later.

Changing the depth threshold from 3 meters to 1 meter will configure Glimpse to trigger only when the wearer's hand is in the field of view (e.g., detecting if the wearer is manipulating objects). These two cascades are evaluated in Section 7.

Similarly, Figure 5b illustrates another cascade that programs Glimpse to trigger whenever the wearer is in a conversation with another human. The steps are identical to the first cascade except here we replace "motion detection" step with "audio detection" which uses a low-power microphone and a microcontroller to detect if there is a conversation happening. A noteworthy point about this cascade is that the "speech detection" feature calculation library function consists of a simple linear classifier coupled with a hardware implementation of spectral features suitable for voice classification tasks [11, 32], yielding useful classification at the 1-mW power level.

We believe the Glimpse architecture attains a useful balance of simplicity, flexibility and safety. By providing optimized libraries that perform most of the heavy lifting, Glimpse makes programming simple. By allowing applications to nevertheless combine a variety of feature libraries in different orders, and by allowing early exits from processing these libraries, Glimpse programs stay expressive and fast. Finally, since applications can only specify linear orderings in which features may be calculated, it is relatively simple to bound the runtime cost of execution of cascades. In particular, because rejection cascades contain no application-level loops, their safety is relatively easy to ensure.

4. COARSE IMAGE PROCESSING

Early-discard systems such as Glimpse have different accuracy requirements from primary imaging pipelines. In particular, primary imagers must discriminate between several classes of interest, and achieve low false-positive *and* false-negative rates for each class. Early-discard systems, on the other hand, typically discriminate between a "background" or "uninteresting" class and interesting ones, often a task requiring less discriminatory power. More importantly, because the primary pipeline will re-process their results, they are only required to have low false negative rates (i.e., they should rarely reject interesting frames) and are allowed modest false-positive rates: as long as they let through only a small number of frames, it is acceptable for some large fraction of those frames to be uninteresting.

Glimpse exploits these less-stringent accuracy requirements by providing, via feature calculation libraries, *coarse* variants of several vision algorithms that are optimized for low false-negative but modest false-positive classification. We have found that *it is feasible to approximate many existing vision algorithms by systematically relaxing traditional computer vision design choices* such as distance metrics, accuracy, data representation bitwidth, output precision, trading off global for local search and "giving up" conservatively on hard cases. In the other words, we show that our approximated algorithms are not only more power efficient than their fine-grained variants, but they also provide satisfactory accuracy when coarser levels of granularity are sufficient. Below, we present two detailed case studies where we use such coarsening techniques, stereo-based depth estimation, and tracking in thermal video. We validate our design choices via measurement studies in Section 6.

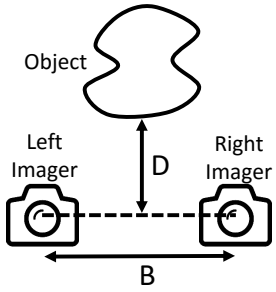
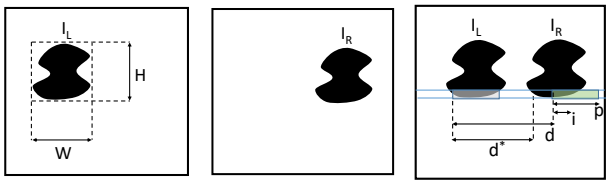


Figure 6: **Stereo vision setup.** An object at distance D in front of a stereo pair of cameras.



(a) left image (b) right image (c) superimposed

Figure 7: **Components of stereo matching.**

4.1 Coarse Stereo on FPGA

As illustrated in Figure 7, traditional stereo takes as input a pair of synchronized images ("left" and "right" images, Figures 7a and 7b) captured by two aligned imagers ("Left Imager" and "Right Imager" in Figure 6) at a fixed baseline distance B from each other. When the left and right images are superimposed (Figure 7c), the image I_L of a given rigid object in the left image is displaced from its image I_R by an x-distance of d^* pixels. The actual depth D of the object is inversely proportional to d^*, so that finding the displacement d^* between corresponding patches in an image (the "correspondence problem") is the central computation in stereo vision [18].

For each original patch at pixel (x, y) in the left image, say the gray length-p patch in Figure 7c, the check is performed by iterating over every length-p *candidate* patch in the corresponding row of the right image (say the relative offset, or *disparity* of the new patch is d pixels), calculating a patch difference metric Δ between the candidate patch (e.g., the green one in Figure 7c) and the original one, and finding disparity d^* that minimizes Δ:

$$d^*(x,y) = \arg\min_d \Delta(I_L(x : x+p, y), I_R(x+d : x+d+p, y)) \quad (1)$$

We optimize this calculation in several ways to reduce power and real-estate consumption on our FPGA. Our goal is a *coarse* depth algorithm that (a) only reports a few discrete depth levels (e.g., 1m, 3m and ∞), and (b) only detects objects greater than a minimum size up to a minimum distance (e.g., size 10cm up to 3m).

4.1.1 Relaxed Distance Metric

Glimpse relaxes the distance measure Δ. The standard measure sum-of-squares (SOS). We instead use the sum of absolute difference (SAD) [26]:

$$\Delta(I_L(x : x+p, y), I_R(x+d : x+d+p, y)) = \sum_{i=1}^{p} |I_L(x+i, y) - I_R((x+i+d), y)| \quad (2)$$

SAD can be implemented in FPGA fabric using a subtrac-

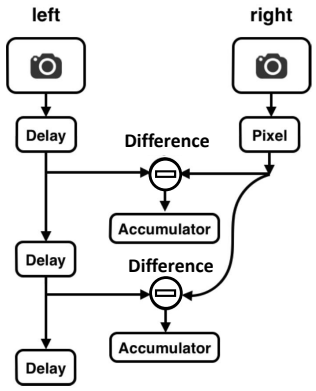


Figure 8: **Microarchitectural support for buffer-free disparity checking.**

tion followed by a comparison, and is significantly less complex than SOS, which requires multiplications.

4.1.2 *Buffer-Free Microarchitecture*

Most imagers output the pixels of each frame row-by-row. Traditional stereo algorithms read the entire frame out to memory (typically DRAM) and then iterate on this buffered data when solving Equation 1. Such a buffered implementation requires almost continuous access to on-board DRAM, adding a significant overhead to the stereo subsystem power draw. To eliminate the overhead of buffering in memory, we implement the correspondence algorithm in a streaming fashion.

The key observation is that finding the correspondence between two rows of spatially separated pixels translates to finding the correspondence between delayed variants of the pixels as they stream out from the imagers. Figure 8 shows the hardware architecture of the depth calculator design. This is composed of three HW building blocks: delay, computing absolute difference between pixels, and accumulator, which adds in and stores the running sum.

Delays are used to shift the pixel stream coming from the left image sensor by the various disparity levels d we wish to try. The delayed pixels from the left image are differenced with the pixels from the right image. Each accumulator then aggregates the results and outputs the measure of similarity for the two image windows associated with a particular delay length. The optimal disparity for a particular window is the delay value which minimizes the similarity measure.

An important implication of the above design is that every iteration of the minimization loop over d in Equation 1 has corresponding circuitry on the FPGA (i.e., the loop is unrolled). Further, within the accumulator, the area of SAD circuitry itself is proportional to the bitwidth of the pixel values being SAD'ed. In addition to avoid buffers, this design has the advantage that these operations can be parallelized, but poses the challenge of requiring additional real-estate.

4.1.3 *Low-Bitwidth Pixel Representation*

Most off-the-shelf cameras use 7 or 8 bits to represent monochrome pixels. We hypothesize that *if only coarse depth estimates are required*, it is possible to truncate the least significant bits (LSB) of the pixels $I(x, y)$ while introducing little error. Note that a few bits of truncation makes the resulting picture indistinguishable to the human eyes from the original. On the other hand, in an FPGA implementation, reducing the bitwidth of numbers manipulated in the innermost loop yields roughly proportional reductions in gate count and power at little reduction in accuracy of coarse depth estimation (Sections 6.1 and 6.2).

4.1.4 *Sparse Disparity Checking*

Equation 1 moves the candidate patch to every possible disparity d with respect to the original patch. In our subsampled VGA image, we must consider $n_d = 60$ different disparity values. The power consumption (and in our FPGA implementation, the circuit size) is directly proportional to n_d. Reducing n_d could therefore substantially improve matching cost.

The intuition behind considering every possible disparity d is to find the d^* that absolutely minimizes the patch difference Δ. However, if we are willing to settle for a difference d^+ that is "close enough" to d^*, we could possibly get away with testing fewer disparities. An initial try may be to shift the candidate patch by $k > 1$ (instead of $k = 1$) pixels when searching for d^+. This brings up the question of how big k can be. If it is too big, we run the risk of not detecting objects smaller than k in image space. If too small, we lose efficiency.

Our approach is to require only that objects of width $\geq W_{min}$ will be detected. For instance, we may specify that the object must be at least the size of a face, e.g., $W_{min} =$ 25cm. It is well known [18] (and simple to verify via high-school geometry) that the width in pixels W_D of an object's image is linearly related to the disparity d^* between its left and right images: $W_D = ad^* + b$ for constants a and b. Intuitively, the closer the object to the stereo pair, the larger the disparity d^*, and the larger the size of the image, so that the disparity and image size vary together. Now if we only require that the candidate patch *overlaps* by a factor of ω to its corresponding image from the other imager, as d increases, the patch expands linearly so that the set of pixels at which the candidate patch can be placed expands: the gap k between positions of the candidate patch (i.e., values of d) can increase with d instead of remaining fixed.

In Glimpse, we fit the values of k greedily using data "labeled" by the brute-force $k = 1$ algorithm. Instead of checking 60 disparities for every original patch, we settle on checking 18 disparities d: {1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15, 17, 20, 23, 26, 30, 34, 39, 44, 50, 57} (thus k increases from 1 to 7), a 70% reduction in matching overhead and circuitry. We require overlap $\omega = 0.95$, and require faces to be detectable: $W_{min} =$ 25cm. Section 6.1 shows that sparse checking is accurate when applied to coarse depth estimation.

4.1.5 *Depth Quantization*

Coarse stereo introduces noticeable error in depth estimates. This is problematic if applications rely on the coarse depth estimation module for fine-grained depth information. For instance, assume coarse depth estimation may introduce an error of 0.3m when the object is about 5m away. If an application were only interested in objects in a band 5.0-5.1m away, using coarse depth to select frames for this purpose could result in mistakenly ignored objects.

To remain conservative, i.e., to maintain good recall, Glimpse's coarse depth API only reports depth in a limited number of bins (e.g., less than 1m, 1-3m, 3-10m and beyond 10m). Given disparity d^+ we derive the corresponding object depth D^+ and quantize it into these bins. The additional quantiza-

tion circuitry is minimal. The big impact, as we show in our evaluation (Section 6.1) is that although the optimizations described in the previous subsection introduce noticeable error in *raw* depth estimation, they are usually well within the slack provided by the *quantized* bins.

4.2 Coarse FIR-Based Tracking

We now turn to functionality based on the thermal (or Far Infra-Red (FIR)) imager, a sensor not broadly available on camera platforms. FIR imagers have lower resolution (16×16-80×80 pixels are common) than RGB, but provide temperature at every pixel. In principle, therefore, they can often directly detect the presence of exceptionally cool or warm bodies (e.g., humans, vehicles, machines, monitors, lighting fixtures) in the environment. One simple but useful feature calculator based on the thermal sensor that Glimpse therefore provides, is a (time-averaged) temperature-based segmentation module, which detects boxes in the field of view with temperature above or below a specified threshold.

In this section, however, we present a related feature calculator, a *tracker*, implemented on the Glimpse microcontroller, given the relatively small volume of data from the FIR imager, that detects and tracks across frames objects in a particular temperature range. Tracking is especially important in early-discard systems such as Glimpse because for many applications, once an object of interest is detected in the field of view, it ceases being of interest. For instance, an application may wish to be notified of a new face coming into the field of view (it may capture a high-resolution frame of the face using the primary imager for further analysis), but once the face is identified it may wish to ignore it.

Glimpse allows high-level applications to notify it to ignore the entity in a particular window. Glimpse does so by tracking the window and not enabling the primary imager on re-detecting it, even if it is otherwise interesting. Prior work [9] has noted a similar benefit in tracking on mobile devices to avoid uploading to the cloud.

4.2.1 Traditional Tracking in RGB

At a high level, tracking in RGB video works as follows [17, 38, 42]. Suppose a window w_t of frame f_t is to be tracked. Given the next frame f_{t+1}, the goal is to calculate window w_{t+1} in f_{t+1}, such that w_{t+1} matches w_t in appearance and dynamics. To perform this match, the tracker may recursively maintain velocity v_t of w_t, hypothesize that the new location of the window is $w'_{t+1} = w_t + v_t$, and search the vicinity of w'_{t+1} for the window w_{t+1} with the best appearance match to w_t. Then the velocity gets updated and the whole process repeated with the next frame, f_{t+2}.

A key design choice is the manner in which the search for the window with the best appearance match is performed. Typically (e.g., [38]), the tracker identifies a set of *keypoints* (locations) in w_t. Each keypoint is associated with a *descriptor* that encodes its surroundings in an illumination (and often scale) invariant manner. The tracker then identifies all keypoints and associated descriptors in f_{t+1}, and matches the old descriptors to a subset of the new ones to determine an appearance match to w_t. Keypoints and related descriptors are relatively computationally intensive to find because they require scanning an entire (relatively) high-resolution image, and need to be illumination invariant since illumination can change significantly across frames in RGB video.

A second choice that adds to the computational overhead of conventional tracking is the manner in which multiple possible matches are resolved, especially when tracking multiple objects. Resolving this ambiguity usually requires some amount of global reasoning [42] possibly including geometry-based consensus [38]. Below, we describe designs for temperature-based keypoints; descriptors that are simple, but fast and adequately accurate by conservatively and quickly handle ambiguities in matching.

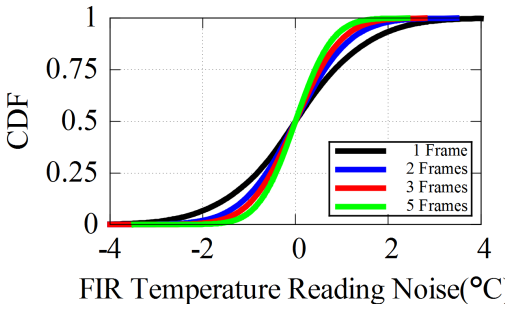


Figure 9: **White noise on the FIR camera's pixel reading.** CDF of noise at each pixel of FIR camera for different frame averaging lengths.

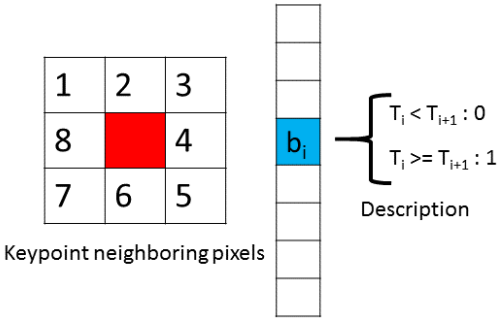


Figure 10: **8-bit binary spiral descriptor.**

4.2.2 Temperature-Thresholded Keypoints

An advantage of FIR imaging is that it is fairly robust to the environment, unlike RGB imaging. The measured temperature of a point does not change due to change in direction of viewing, illumination sources in the environment, shadows, etc. Given that the tracker is intended for use to track objects that have temperature T significantly different from the background, Glimpse therefore uses the "raw" temperature to find candidate locations for keypoints: all pixels in some temperature range $T \pm \delta$ (we pick $\delta = 3°C$ currently) are considered potentially matching keypoints. Given that the FIR imager has only 1024 pixels (significantly less pixels than a regular RGB camera), keypoint generation is extremely fast.

One complication is that FIR imagers are prone to (zero-mean Gaussian) white noise [12]. We therefore average across $n = 3$ frames to reduce the temperature variance. Figure 9 shows CDF of pixel reading noise from a fixed scene when averaged over frame sub-sequences of varying lengths. It is clear that no averaging (black curve) will result in about $\pm 4°C$ of noise, but a moving average over three consecutive frames will reduce the generated noise to about $\pm 2°C$. Although averaging over five frames reduces the noise even further, the resulting blur is unacceptable for our purposes.

4.2.3 Temperature-Spiral Descriptors

Given keypoints, we seek to associate descriptors with them

in order to determine which, if any, match with descriptors from the previous frame. To ensure illumination, rotation and scale invariance, traditional descriptors are fairly computationally intensive, involving multiple rounds of gradient, histogram and multi-scale smoothing calculations [10, 31].

For our descriptors, we simply choose an encoding of the temperature values around the keypoint arranged in a spiral. We present four different encodings/descriptors here and evaluate their performance later in this paper. The descriptors, detailed below, are **1.** 8-bit binary **2.** 24-bit binary **3.** 8-point absolute temperature and **4.** 24-point absolute temperature.

Figure 10 shows the 8-bit binary descriptor. The keypoint (red pixel) and its neighboring pixels are shown in the Figure 10. This descriptor is an 8-bit number such that each of its bits (b_i) represent the logical result of comparing temperature at pixel i and $i+1$. Similarly, 24-bit binary descriptor follows the same procedure with the difference that we consider two layers of circular neighboring pixels around the keypoint instead of just one. The 8 and 24-points absolute temperature descriptors are similar to the binary versions with the difference that the i^{th} element of the descriptor vector is the temperature of the i^{th} neighboring pixel of the corresponding keypoint.

Finding the best match is standard: we use the sum of absolute differences to compare absolute temperature descriptors and using bitwise-xor for binary descriptors.

4.2.4 *Losing Track Conservatively*

A limitation of FIR imaging is that its resolution is low, and different instances of objects often have indistinguishable thermal signatures. For instance, it is infeasible (for even humans) to tell one person from another using currently available low-power thermal imagers. The problem of resolving ambiguities between descriptors from multiple objects is therefore especially challenging in FIR tracking.

Our solution is simply to be conservative about matching. Applications will be notified if there is a potential break in the track and it is up to the application to stitch together tracks, potentially using higher level information such as RGB pixels.

To implement conservative matching, at each time step, Glimpse first expands each keypoint in the new frame f_{t+1} into a "blob" (i.e., connected component of similar temperature values) surrounding it by iteratively thresholding on pixel values, a simplified version of Otsu's segmentation method [39]. If any pair of blobs in the new frame are "too close to each other", we terminate the track.We determine if two blobs are close to each other by dilating them by a single pixel [2] and checking if the blobs merged. Given that iterative thresholding, connected components and dilation are fast operations, the implementation is fast.

5. HARDWARE IMPLEMENTATION

Glimpse is fully implemented as a modular add-on board (1a) called the "Glimpse Gating Board" (GGB). The board is designed to run autonomously for a day on a 200mAh battery. Any primary imaging system that has an "image enable" line can be connected to Glimpse via a control line and can communicate with it via a 2 Mbps SPI bus.

[2]Note that at 32×32 resolution, a single pixel represents several degrees of spatial separation.

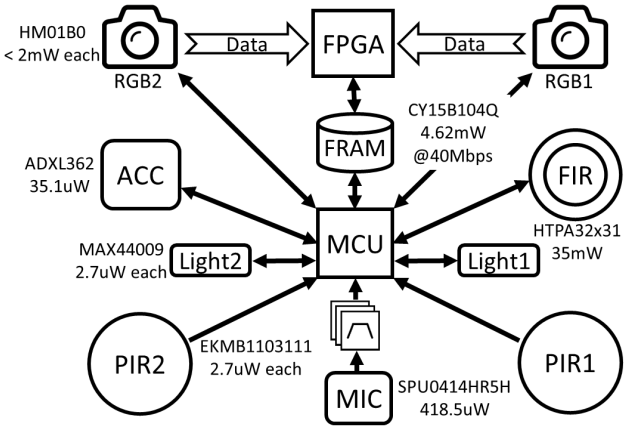


Figure 11: **Glimpse gating board implementation block diagram.**

Figure 11 shows the block diagram for GGB. Given our small power budget, all components are selected to balance low power while providing adequate performance for coarse, cascaded algorithms. Both the (Ti MSP430) microcontroller and the (AGLN250 Igloo Nano) FPGA are some of the lowest-power parts available in their category. Even a careful implementation of coarse stereo matching (along with video readout circuitry for the two imagers), for instance, consumes half the FPGA. In many cases, power consumption of components will reduce significantly in the near future, tracking advances in commercially available parts. For instance, that the highest power consumer is the (relatively low-resolution) thermal FIR imager, a part ripe for improvement.

6. EVALUATING COARSE VISION

We wish to answer three questions about the coarse vision (depth estimation and stereo tracking) algorithms introduced in Section 4:

- What is their accuracy?
- What is their resource consumption?
- How do these vary with our design choices?

We use data collected under controlled conditions in these experiments. We restrict detailed experimental evaluation to the low-bitwidth representation, sparse disparity checking and depth quantization optimizations of Section 4. With respect to the other optimizations, briefly: 1. Using SAD instead of SOS for distance measure (Section 4.1.1) yields a depth measurement circuit that uses 42% fewer gates[3], and 2. Buffer-free implementation of stereo (Section 4.1.2) saves 13% in average power draw over the buffered version.

6.1 Coarse Stereo: Accuracy

What is the impact of reducing the number of bits used to represent pixels (Section 4.1.3) on precision of depth measurement? To answer this question, we placed the Glimpse camera in front of a textured wall and moved the camera between 0.5m to 3.9m from the wall, by 0.3m increments. For a set of patches of varying sizes $s \times s$ (s selected from 20, 40, 60 and 100pixels) in the field of view, we calculated

[3]Power reductions should be similar. The SOS-based circuit is too large for implementation on our FPGA, so we cannot report measured power gains.

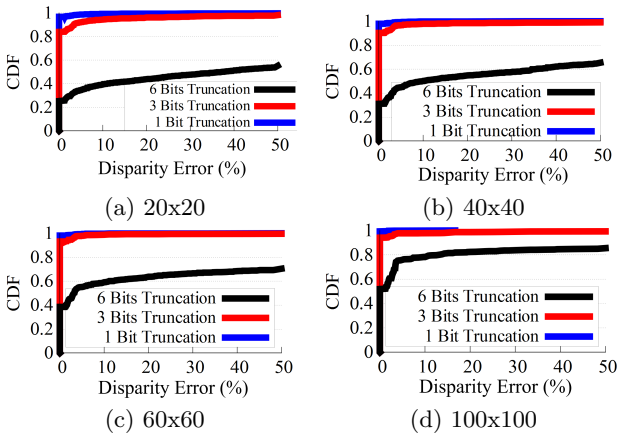


Figure 12: **Disparity error versus truncation length.**

the pixel disparity d^* while varying the number b of bits used to represent the captured stereo pairs. We used b = 7, 6, 4 and 1, corresponding to 0, 1, 3 and 6-bit truncation. Figure 12 reports the CDF of error in disparity d^* relative to the disparity measured from the un-truncated stereo-pair version.

Two points are worth noting. First, relative error with a 3-bit truncation (i.e., b=4) is quite acceptable, whereas going down to 6-bit truncation introduces a significant amount of error in disparity estimation. We therefore use 3-bit truncation in Glimpse. Second, although the error depends to some extent on the size of the patch being matched (larger patches contain more visual information and thus less matching error), even small patches can be matched with quite small error with 3 bits truncated. Even with s = 20-sized patches, relative disparity is zero 82% of the time (red line in 12(a)) and 94% of the time, it is less than 9%. With s = 60, disparity is zero over 93% of the time.

We now study the impact of quantized depth levels on the accuracy of depth estimation Section 4.1.5. Suppose that instead of estimating depth as precisely as possible, we are only interested in knowing which of four bins the depth belongs to: bin_0:0-1m, bin_1: 1-2m, bin_2: 2-3m or bin_3: 3-∞m. We focus now on the challenging 20×20 patches and quantize the results of the above experiment into these bins. In other words, we convert the pixel disparity for each measurement into the actual depth then instead of basing our analysis on the actual depth, we measure the bin index in which it falls into. Figure 13 shows the results when 1 to 6 bits are truncated. For each truncation level, the bar shows the distribution of the *bin error*. The bin error is simply the absolute value of the difference between bin index calculated using the truncated and un-truncated data. Focusing on the 3-bit truncation case, we see that *at the bin level*, 96% of the measurements have zero bin error, and almost all the rest are off by just one bin. We also note in passing that even for 6-bit truncation (i.e., just black and white pixels), quantized accuracy is encouraging: ~60% of measurements are binned correctly.

In the above algorithms, we calculated the optimal disparity d^* by exhaustively checking all 60 possible disparities. We now evaluate *sparse disparity checking* (from Section 4.1.4), assuming that a) we are only interested in depth of objects wider than 25cm, and b) we are only interested in bin-level precision. We recomputed bin-level results as in the previous experiment, but now require the algorithm to

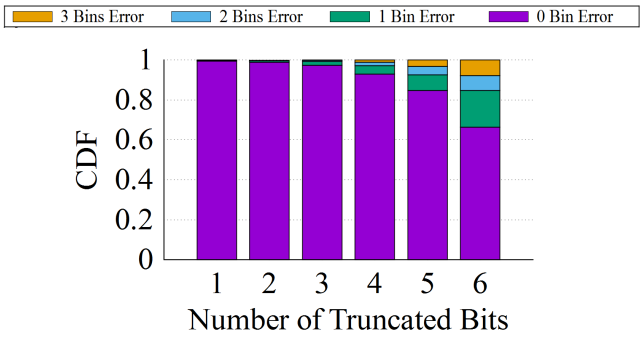


Figure 13: **Bin-level depth error vs. truncation.**

use just $n \in \{7, 17, 22, 37, 41, 50\}$ different disparity checks, using fixed bitwidth b = 4 (i.e., truncation length 3). We compare the binning error relative to using all 60 checks. Figure 14 shows results for patches of sizes s = 20 and 100. The upshot is that $n \approx 37$ checks are enough for excellent bin-level accuracy over all patch sizes. In Glimpse, because our patches tend toward being larger, we use just n = 22 disparity checks.

6.2 Coarse Stereo: Resource Consumption

How does resource consumption vary with our approximate implementation of depth detection? Our primary target FPGA is the Igloo Nano AGLN250. However the AGLN250 is not big enough to handle stereo depth detection with full disparity checks and un-truncated pixels. For comparison purposes, we therefore consider the more powerful (and power-hungry) AGL400 and M1AGL1000 FPGAs as well.

In our first experiment, we fixed bitwidth at 7, increased the total number of allowed disparity checks until the AGLN250 ran out of resources, switched to AGL400 and continued increasing the number of disparity checks until it ran out of resources too and finally switched to M1AGL1000. At each point we let the FPGA calculate the depth map over the entire VGA frame which is computationally the most demanding version of our depth detection algorithm. We then simulate power consumption and resource utilization of the FPGA using Microsemi Libero SoC software. Figure 15a shows power consumption and Figure 15b shows resource utilization (i.e. fraction of all available logic elements on the FPGA used) of the three FPGAs under different pixel disparities. As shown in the figures, the power consumption increases generally as the number of disparity checks increases. Also the utilization of a particular FPGA increases as the number of disparity checks increases.

Figure 16 examines the power draw and utilization impact of different bit truncation. These experiments are performed with n = 22 disparity checks. Resource use drops almost linearly with optimization level (truncation length in this case). Comparing with Figure 13 and Figure 14, we see

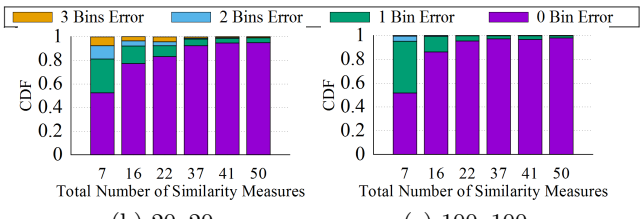
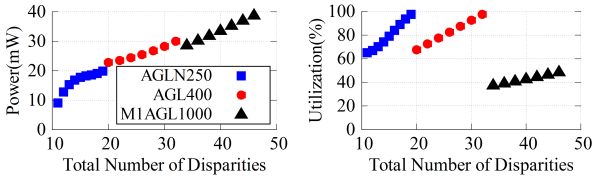
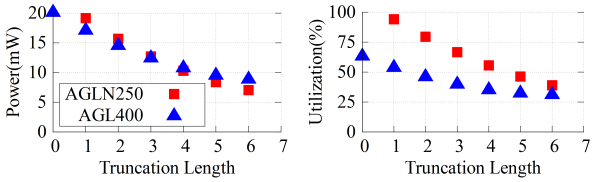


Figure 14: **Bin-level error vs. no. of disparity checks.**



(a) Power Consumption (b) Gate Utilization

Figure 15: **Resource use versus disparity checks.**



(a) Power Consumption (b) Resource Utilization

Figure 16: **Resource use versus pixel truncation.**

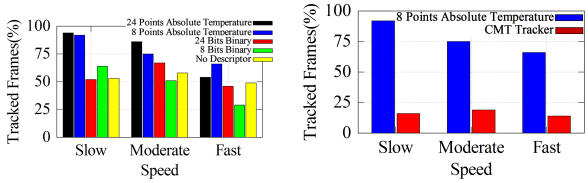
that 3-bit truncation and 22 disparity checks is likely the accuracy/power sweet spot, yielding 95%+ accuracy on size-60 patches at a stereo-computation draw of $\sim 12mW$ while using less than 75% of the AGLN250 logic elements.

6.3 FIR Tracking

We now evaluate the coarse FIR tracking algorithm. We try to answer three questions: 1. **Accuracy:** How well does the algorithm keep track? 2. **Resource use:** How much energy does the algorithm consume? 3. **Design choices:** How well do the various spiral descriptors work?

To answer these questions, we performed the following experiment. We pointed Glimpse into an empty office room and asked two human subjects to enter and interact in the field of view in a scripted manner for up to 25 seconds per interaction. We collected 7 versions of the interaction where subjects were moving at "slow" (leisurely strolling with stops), "medium" (walking) and "fast" (brisk walking, almost running) speeds. We manually annotated the tracks i.e., added distinct boxes surrounding each subject in the video and compared these "ground-truth" tracks to those returned by (variants of) our algorithm.

Figure 17a shows how the average fraction of properly tracked frames over total available frames varies with the speed of movement and the various descriptors (in the "no descriptor" case, just the temperature of the central pixel is used as descriptor). The noteworthy points are: First, absolute temperature descriptors are much better than the binary ones; the absolute value of the temperature is clearly a lot more informative than a single bit representation *temperature change* at each pixel. Second, the difference in tracking quality between the 8-point and 24-point descriptors is relatively small, encouraging the use of 8-point descriptors due to their lower computational complexity. Third, given the absolute descriptors, the tracker does an acceptable job of tracking: it maintains track for 60-90% of tracks on average, depending on whether subjects are moving fast or slow. Fourth (Figure 17b), we applied a tuned state-of-the-art RGB tracker [38] to the FIR videos to evaluate the importance of FIR-specific descriptors. 17b shows that the FIR-specific tracker clearly does better; we found that standard RGB descriptors do not fare well with low- resolution thermal footage.



(a) Comparison of descriptors (b) FIR vs RGB tracker

Figure 17: **FIR tracking evaluation.** 17a shows for how many frames our FIR tracker can contiuously track an object given different descriptors, and 17b compares the track length of our FIR tracking algorithm vs a well-known object tracking algorithm when applied to a theremal video.

Scenario	PIR	FIR	Depth	Total	Precision
Driving	0.96	0.30	0.43	0.12	31%
Corridor	0.94	0.00	0.33	0.00	100%
Monitor	0.83	0.07	0.34	0.02	8%
Outdoor	0.99	0.23	0.20	0.04	1.7%
Lunch	0.99	0.50	0.34	0.17	42%
Meeting	0.97	0.13	0.23	0.03	12%
1 to 1	0.99	0.09	0.31	0.03	36%

Table 1: **Glimpse pass-through rates and precision.**

7. END TO END EVALUATION

Finally, we wish to understand how well suited the Glimpse system is to its primary role of supporting day-long wearable vision. One of the authors wore a Glimpse-based camera for 15 hours over three days of daily activity (roughly 3, 7 and 5 hours each) while running a cascade (Figure 5a) on Glimpse. Glimpse produced a high-resolution 30-frame video snippet every time a person came within 3 meters from the wearer (typically a conversation partner) or the wearer's hand entered the Glimpse's field of view. The rejection cascade used is similar to that in 5a, except that the thermal segmentation module was replaced by an FIR tracker as in Section 4.2. The analysis of another cascade that triggers the primary imager only when the wearer's hand appeared in the field of view is presented in Section 7.3.

All video produced by the primary imager was offloaded via wireless to a mobile device. In this experiment we did not attempt off-board analysis of the video. Common activities performed by the wearer included but were not limited to, *driving*, *going for lunch with friends*, *one-to-one meeting*, *group meeting*, *playing ping-pong*, *corridor walking*, *outdoor walking*, and *working in front of a monitor*.

We paired up a Glimpse Gating Board (GGB) with a Raspberry Pi 3 board (which we call *Gated Pi* below) as the primary imager, via a trigger wire. The Pi has an 8MP imager. For the mobile device, the wearer carried a WiFi connected laptop in a backpack; a production system would likely use a mobile phone instead. To collect ground truth, we added a second Raspberry Pi (the *Continuous Pi*) that recorded *all* video in the field of view. Both Pis offloaded data to the laptop via WiFi, one sporadically, the other continuously. We logged *real-time* power draw of the GGB, the Gated Pi and its components to the laptop via a National Instrument data acquisition system in the backpack.

Event Duration, d (s)	$d \geq 3$	$3 > d \geq 2$	$2 > d \geq 1$
Recall Rate (%)	100	96	86.5

Table 2: **Glimpse recall rate vs event duration.**

7.1 Gating Accuracy

The primary job of Glimpse is to reject many uninteresting frames while letting through almost all interesting ones. In daily use, what fraction ("pass-through rate") of frames did it allow to pass through? What fraction of these truly should have passed through ("precision")? What fraction of all interesting events actually went through ("recall")? We answer these questions in this section.

Our detailed power measurements allowed us to monitor in real time when each component of each cascade triggered, and whether or not the Gated Pi was finally triggered. We were therefore able to measure the pass-through rate of each stage of the rejection cascades, and of the two cascades as a whole. Table 1 reports the results, broken down per scenario. For each cascade stage, we report the fraction of *its* input passed through.

Overall, total pass-through rates range from 0 to 17%, with an average (not shown) of roughly 6%. The ultra-low-power PIR can reject 1-17% of input: it is especially good at detecting inactivity when sitting roughly still in front of the monitor. Both FIR and Depth are highly effective gates. When walking in the corridor on these three days, the wearer encountered just two persons, for which the FIR stage let through 3 frames, which rounds off to a 0.00% pass-through. The Depth system allowed one of these frames, for a 1/3 pass-through rate.

We inspected all passed through frames manually to determine what fraction had wearer hands or nearby people i.e., the precision. As the "Precision" column shows, Glimpse's precision is modest. Again, since the one frame let through in the corridor was indeed a person, precision is 100%, even though the overall pass-through rate rounds to 0.00.

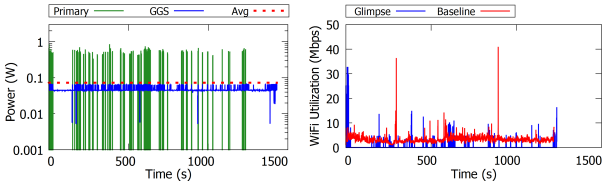
Measuring recall is harder, since we need to inspect roughly 1.6M frames from Continuous Pi to determine if we missed any events. We looked through the video manually and counted instances where a hand or person appeared in the field of view for duration d seconds, such that $d \geq 3$, $3 > d \geq 2$ and $2 > d \geq 1$. We found the shorter the event, the more episodes there were when the object moved so fast through the field of view that GGS, at 10FPS, was unable to detect it. For events of modest (e.g., > 2s) length, however, Glimpse has excellent recall (Table 2).

7.2 Resource Usage

What is Glimpse's average power and wireless bandwidth consumption? The low overall pass-through rate (6%) indicates that power consumed by the primary imaging pipeline, and data sent by it, should be correspondingly low. Further, by design, the Glimpse board itself should have low power draw. How do these play out in practice?

Scenario	Total	Primary	GGS	Depth	FIR
Driving	116	67	49	6	38
Corridor	43	0.6	42	0.2	37
Monitor	80	39	41	1.4	33
Outdoor	80	30	50	5.1	40
Lunch	153	99	54	10	39
Meeting	85	39	46	2.8	38
1 to 1	73	26	47	2.2	39

Table 3: **Glimpse power consumption (mW).**



(a) Real-time Power Consumption (b) Real-time WiFi Utilization

Figure 18: **Real-time Power and WiFi Utilization.** Power consumption and WiFi transmissions of a primary imager become bursty when it is being gated by Glimpse system during an example one-to-one meeting scenario.

Table 3 shows the average power (mW) breakdown for the major components of GGS, that of Gated Pi and their sum. Overall average for the total power consumption (not shown in the table) was roughly **112mW**. 18a shows the real-time power traces for Glimpse system in blue, Gated Pi in green, and their overall power consumption in red. This curve illustrates that power consumption of the Gated Pi becomes bursty in the existence of Glimpse system in a sample scenario of one-to-one. For comparison, the average power draw of Continuous Pi, which off-boards all data is **1016mW**. A few points are worth noting. First, Glimpse *total* power consumption is low both in absolute and relative terms. In relative terms, it is **7x** to **25x** lower than pure offloading, depending on the scenario. In absolute terms, its *worst-case* 153mW draw will allow it to run for over 5 hours on a tiny 200mAh battery; the average of 112mW would allow almost 7 hours. Second, the gating board itself (average draw 55mW) can easily be powered by a 200mAh battery through a day. Third, the FIR sensor and tracker is a power bottleneck, with the sensor itself averaging well over 20mW.

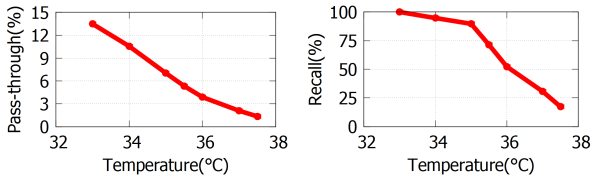
Finally Table 4 shows savings in bandwidth utilization. Figure 18b shows intermittent WiFi transmissions of Gated Pi when triggered by Glimpse system in a sample scenario of one-to-one meeting. Even when Gated Pi transmits a one-second burst of video every time it is triggered, it still sends 4× to 16× less data. Our measurements show that if it only sent one frame each time it was triggered, data sent would reduce by another 3.2×. Overall, therefore, Glimpse clearly has the potential to reduce data transmitted by roughly 12-50× relative to pure offloading. Of course, all these reductions are predicated on the application only being interested in a sparse set of frames.

7.3 Glimpse Programmability

Here we run a cascade on the Glimpse similar to Figure 5a except the depth threshold is set to 1 meter. The goal here is to trigger the primary imager whenever the wearer's hand enters the field of view, indicating that the wearer is probably interacting with an object. Here we ask the Glimpse wearer to interact with objects in hand while standing in an office setting with background thermal activities. In each experiment we varied the temperature threshold from $33°C$ to

Scenario	Gated BW(Mbps)	Baseline BW(Mbps)
Driving	1.01	5.90
Corridor	0	5.15
Monitor	0.58	8.72
Outdoor	2.46	10.56
Lunch	2.61	8.25
Meeting	0.14	1.58
1 to 1	0.24	3.14

Table 4: **Glimpse WiFi bandwidth consumption.**



(a) Pass-through vs Temperature (b) Recall vs Temperature

Figure 19: **Recall vs pass-through rate trade-off.** Recall and trigger dependence on the temperature threshold for hand detection cascade.

$37.5°C$ and measured the recall and the pass-through rates of the detected frames.

Figure 19 shows the recall and pass-through rates versus the threshold temperature on two separate graphs. The key observation is that as the temperature threshold increases the pass-through rate decreases which helps lowering the power consumption but it decreases the recall rate as well. There is a clear trade-off between the total power consumption (pass-through rates) and the resulting recall rate, which can be controlled straightforwardly via the corresponding parameter in the rejection cascade.

8. RELATED WORK

Glimpse is built based on previous research on continuous mobile vision, ultra-low power depth sensing, and FarIR tracking. However, our system differs from previous work from multiple perspectives.

Continuous mobile vision: Prior work investigates various layers across the vision processing pipeline to identify the power bottleneck of each layer. Traditional off-the-shelf image sensors' power consumption is not proportional to their frame rate and resolution, thus it is not feasible to trade quality with power consumption using them. [23] [27] [5] tune hardware parameters of an image sensor to achieve proportional energy consumption to the number of pixels sensed. [29] leverages shared intermediate results within a vision processing software framework to reduce redundant computational overhead. [15] and [6] look at the benefit of using low-power accelerometers to gate the operation of the image sensor. Perhaps the biggest difference between our work and these is our advocacy for a *dedicated* hardware/software subsystem that, perhaps paradoxically, includes imagers, that applications can use in a *programmable* manner to control the computer vision costs they would incur by using the primary imaging pipeline.

Sensing depth at low power: Sensing depth at low power is hard in general. Several depth sensors [2] can operate at low power and high resolution. However, these sensors just report the depth of a single pixel. Other sensors [3, 34, 46, 48, 52, 53] can compute the depth of a wide field of view with high resolution. However, their power consumption is high. Our primary insight is that *for purposes of early discard*, it is often adequate to have depth estimated in subsets of the field of view and at coarse resolution. By identifying corresponding relaxations in the algorithm design, we deliver useful depth estimation at a few mW.

FarIR tracking at low-power: Prior works use FarIR sensors to detect the presence of objects of interests for Heating, Ventilation, Air Conditioning, (HVAC) and lighting control of building [13]. [20] also looks at the opportunity of using FarIR sensors for objects tracking. Different from these work, our FarIR subsystem is a customized design that can run on a microcontroller at low-power consumption.

Battery-free cameras: Work parallel to ours has shown battery-free cameras that harvest energy from ambient Radio Frequency (RF) signal sources and capture and transmit images on a forced duty-cycled basis [36] [45] [37]. These devices reduce the maintenance costs of camera systems by eliminating the battery, but due to their constrained energy budget, they suffer from limited computational capabilities required for on-board complex machine vision operations [35]. Although the vision algorithm presented here cannot be applied to those such energy constrained devices, the idea of application-specific cascaded operation for gating a high-power sensor using low-power sensors and algorithms (such as the use of motion detectors) seems a feasible approach to enable battery-free devices with more continuous operation.

9. CONCLUSION

In this work we designed, implemented and evaluated Glimpse, a programmable embedded system designed to detect at low power events worthy of further processing. Key innovations include a low-power imager array and computational fabric dedicated to image-discard calculations, a safe but expressive programming model for applications and coarse variants of conventional stereo depth and tracking algorithms suited for the discard pipeline. Combining Glimpse with a high-power imaging pipeline system allows the high-power pipeline to process less than 10% of the frames it would have processed otherwise, while drawing about 100mW of overall power. Correspondingly, the amount of video data transmitted for further processing is also lower by an order of magnitude. Based on these results, we believe that Glimpse moves continuous vision on lightweight wearables to the realm of the practical.

References

[1] Ambarella a9 ultra hd 4k camera soc product brief. http://www.ambarella.com/uploads/docs/A9-product-brief.pdf.

[2] Long distance measuring sensor datasheet. http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a02yk_e.pdf. Accessed: May 2017.

[3] Stereographic depth mapping on an fpga. https://courses.cit.cornell.edu/ece576/FinalProjects/f2010/pfk5_jk459/pfk5_jk459/index.html.

[4] S. Agarwal, M. Philipose, and P. Bahl. Vision: The case for cellular small cells for cloudlets. In *International Workshop on Mobile Cloud Computing and Services*, 2014.

[5] S. U. Ay. A 1.32 pw/frameÂě pixel 1.2 v cmos energy-harvesting and imaging (ehi) aps imager. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pages 116–118. IEEE, 2011.

[6] P. Bahl, M. Philipose, and L. Zhong. Vision: cloud-powered sight for all: showing the cloud what you see. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 53–60. ACM, 2012.

[7] S. Bambach, J. M. Franchak, D. J. Crandall, and C. Yu. Detecting hands in childrenâĂŹs egocentric views to understand embodied attention during social interaction. *Proceedings of the 36th Annual Conference of the Cognitive Science Society (pp. 134-139). Quebec City, Canda: Cognitive Science Society*, pages 134–139, 2014.

[8] S. Blessenohl, C. Morrison, A. Criminisi, and J. Shotton. Improving indoor mobility of the visually impaired with depth-based spatial sound. In *ICCV-ACVR workshop*, December 2015.

[9] T. Y. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys 2015, Seoul, South Korea, November 1-4, 2015*, pages 155–168, 2015.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[11] Y. Deng, S. Chakrabartty, and G. Cauwenberghs. Analog auditory perception model for robust speech recognition. In *IEEE International Joint Conference on Neural Networks*, 2004.

[12] E. L. Dereniak and G. D. Boreman. *Infared Detectors and Systems*. Wiley, second edition, 1996.

[13] V. L. Erickson, A. Beltran, D. A. Winkler, N. P. Esfahani, J. R. Lusby, and A. E. Cerpa. Toss: Thermal occupancy sensing system. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–2. ACM, 2013.

[14] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards wearable cognitive assistance. In *MobiSys*, 2014.

[15] S. Han, R. Nandakumar, M. Philipose, A. Krishnamurthy, and D. Wetherall. Glimpsedata: Towards continuous vision-based personal analytics. In *Proceedings of the 2014 workshop on physical analytics*, pages 31–36. ACM, 2014.

[16] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. MCDNN:An Approximation-Based Execution Framework for Deep Stream Processing Under Resource Constraints. In *Proceedings of the 14th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 2016.

[17] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M. M. Cheng, S. L. Hicks, and P. H. S. Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, Oct 2016.

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[19] J. Healey and R. W. Picard. Startlecam: A cybernetic wearable camera. In *Second International Symposium on Wearable Computers (ISWC 1998)*, pages 42–49, 1998.

[20] P. Hevesi, S. Wille, G. Pirkl, N. Wehn, and P. Lukowicz. Monitoring household activities and user location with a cheap, unobtrusive thermal sensor array. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 141–145. ACM, 2014.

[21] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. R. Wood. Sensecam: A retrospective memory aid. In *UbiComp 2006*, pages 177–193, 2006.

[22] J. Hoisko. Context triggered visual episodic memory prosthesis. In *Fourth International Symposium on Wearable Computers (ISWC200*.

[23] K. Kagawau, S. Shishido, M. Nunoshita, and J. Ohta. A 3.6 pw/frameâ· pixel 1.35 v pwm cmos imager with dynamic pixel readout and no static bias current. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 54–595. IEEE, 2008.

[24] T. Kanade and M. Hebert. First-person vision. *Proceedings of the IEEE*, 100(8):2442–2453, 2012.

[25] S. Krishna, G. Little, J. Black, and S. Panchanathan. A wearable face recognition system for individuals with visual impairments. In *Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '05, pages 106–113, New York, NY, USA, 2005. ACM.

[26] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos. Review of stereo vision algorithms: From software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[27] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 69–82. ACM, 2013.

[28] R. LiKamWa, Z. Wang, A. Carroll, F. X. Lin, and L. Zhong. Draining our glass: An energy and heat characterization of google glass. *arXiv preprint arXiv:1404.1320*, 2014.

[29] R. LiKamWa and L. Zhong. Starfish: Efficient concurrency support for computer vision applications. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 213–226. ACM, 2015.

[30] H. Liu, M. Philipose, M. Pettersson, and M.-T. Sun. Recognizing object manipulation activities using depth and visual cues. *Journal of Visual Communication and Image Representation*, 25(4):719–726, 2014.

[31] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[32] H. Lu, A. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu. Speakersense: Energy efficient unobtrusive speaker identification on mobile phones. In *International Conference on Pervasive Computing*, pages 188–205. Springer, 2011.

[33] B. Mandal, S. Chia, L. Li, V. Chandrasekhar, C. Tan, and J. Lim. A wearable face recognition system on google glass for assisting social interactions. In *Computer Vision - ACCV 2014 Workshops - Singapore, Singapore, November 1-2, 2014, Revised Selected Papers, Part III*, pages 419–433, 2014.

[34] C. Murphy, D. Lindquist, A. M. Rynning, T. Cecil, S. Leavitt, and M. L. Chang. Low-cost stereo vision on an fpga. In *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, pages 333–334. IEEE, 2007.

[35] S. Naderiparizi, Z. Kapetanovic, and J. R. Smith. Wispcam: An rf-powered smart camera for machine vision applications. In *Proceedings of the 4th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, pages 19–22. ACM, 2016.

[36] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith. Wispcam: A battery-free rfid camera. In *RFID (RFID), 2015 IEEE International Conference on*, pages 166–173. IEEE, 2015.

[37] S. Naderiparizi, Y. Zhao, J. Youngquist, A. P. Sample, and J. R. Smith. Self-localizing battery-free cameras. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 445–449. ACM, 2015.

[38] G. Nebehay and R. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 862–869, March 2014.

[39] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[40] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015.

[41] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE, 2012.

[42] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1201–1208. IEEE, 2011.

[43] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan. Odessa: Enabling interactive perception applications on mobile devices. In *Mobisys*, 2011.

[44] S. Rallapalli, A. Ganesan, K. Chintalapudi, V. N. Padmanabhan, and L. Qiu. Enabling physical analytics in retail stores using smart glasses. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 115–126. ACM, 2014.

[45] V. Talla, B. Kellogg, B. Ransford, S. Naderiparizi, S. Gollakota, and J. R. Smith. Powering the next billion devices with wi-fi. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, page 4. ACM, 2015.

[46] N. H. Tan, N. H. Hamid, P. Sebastian, and Y. V. Voon. Resource minimization in a real-time depth-map processing system on fpga. In *TENCON 2011-2011 IEEE Region 10 Conference*, pages 706–710. IEEE, 2011.

[47] R. Tapu, B. Mocanu, and T. B. Zaharia. ALICE: A smartphone assistant used to increase the mobility of visual impaired people. *JAISE*, 7(5):659–678, 2015.

[48] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, 1987.

[49] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[50] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg. A scalable approach to activity recognition based on object use. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[51] Z. Xu, M. Kusner, M. Chen, and K. Q. Weinberger. Cost-sensitive tree of classifiers. In *ICML*, 2013.

[52] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[53] Z. Zhang. Camera calibration with one-dimensional objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(7):892–899, 2004.